INSA Strasbourg

# Interface graphique

## Drone Cigogne

**Gaylord Wagner**

**Eté 2009**



## Sommaire

# Introduction

Le but de ce stage est de concevoir une interface graphique pour suivre et contrôler le drone quand il est en vol. Et le tout le plus simplement possible.

Pour ce faire une interface graphique développée par Martin DEHAUT avait été conçue. Cependant certains problèmes sont apparus au fur et à mesure des versions, qui empêchaient l'exécution du code sur certains ordinateurs. De plus, le manque de commentaires dans le code et de rapport sur le code complique extrêmement l'évolution du code par une autre personne.

De ce fait, la nouvelle interface sera développée sous Labview™. D'une part le langage utilisé sous Labview™ (Langage G) est fortement orienté objet ce qui simplifie grandement la tâche de conception d'une interface et deuxièmement le code est facilement portable et modifiable par d'autres personnes (et compréhensible).

Tout cela fait que nous développerons donc sous Labview™.

Dans ce pré rapport je ne vais pas expliquer tout les aspects du langage G. Ceci sera fait dans le rapport final. **C'est une explication succincte.**

# Cahier des charges

L'interface devra comprendre les parties suivantes :

- ✓ Réglage des Gains
- ✓ Réglage des offsets servomoteurs
- ✓ Contrôle des vitesses, altitude et niveau de batterie
- ✓ Visualisation de la trajectoire
- ✓ Envoie d'une trajectoire au drone en vol
- ✓ Configuration de la trame GPS
- ✓ Acquisition des données par le port série

## Réglage des Gains

Le réglage des Gains s'effectue via un écran dédié à ceci :

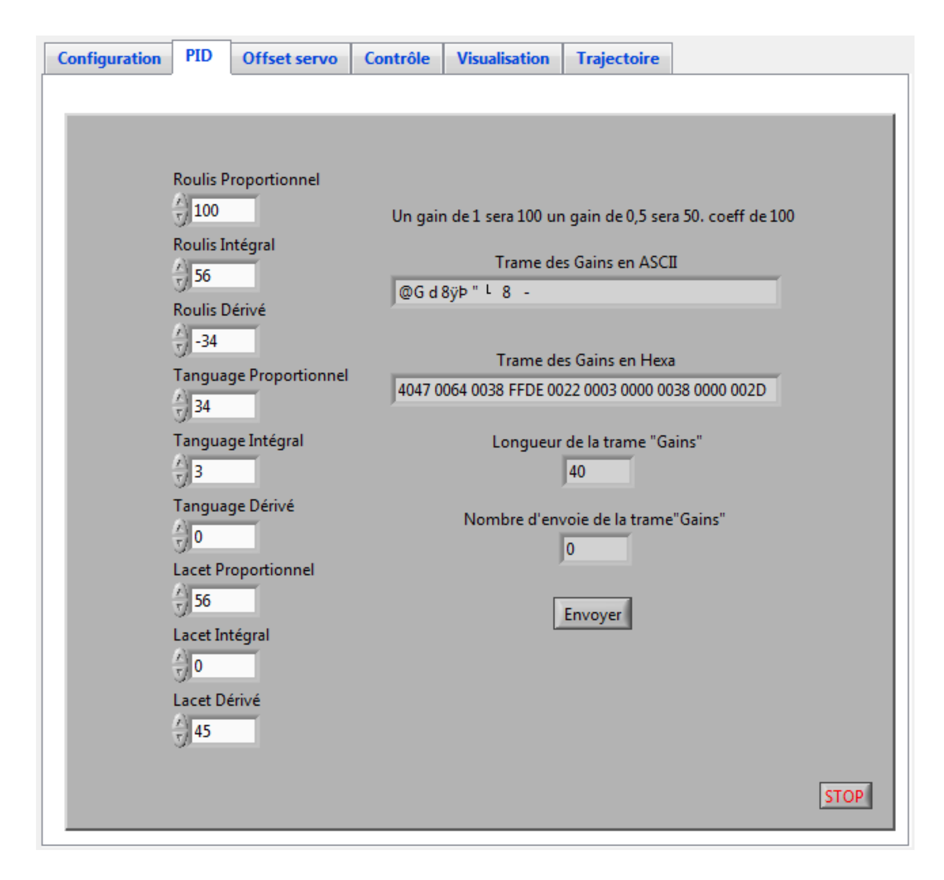



Figure 1 – écran Gains

L'écran permet de régler les différents gains, de visualiser la trame envoyée au format ASCII et la trame envoyée au format hexadécimal. La trame sera vue en hexadécimal par le dsPIC de la carte mère du drone, et sera traité pour en extraire les différentes valeurs. La position des différentes valeurs est fixe dans la trame. Un coefficient correspond à deux caractères ASCII. Et donc, par conversion, à deux octets soit quatre chiffres hexadécimaux.

On y affiche aussi la longueur de la trame et le nombre d'envoie en cours de la trame.

## Explication sommaire du code

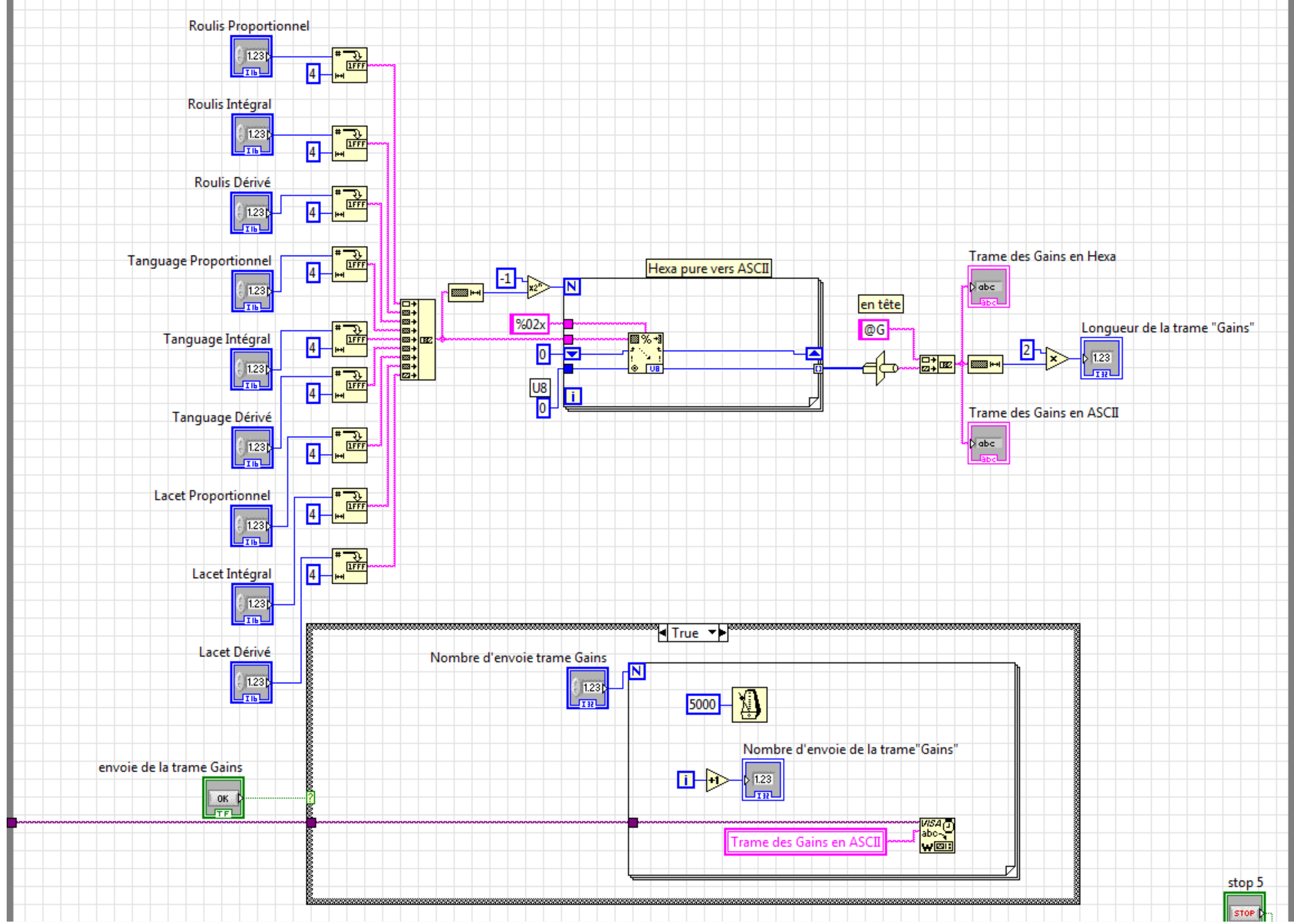



Figure 2 – Langage G des gains

Sur la figure 2 on peut voir le langage G des gains. Dans l'ordre,  on peut y voir les blocs d'acquisitions des valeurs. On prend les valeurs dites « number » que l'on convertit en hexadécimal pure. Ce qui nous des chaines hexadécimal de longueur de quatre caractères. On peut concaténer toutes les chaines entres elles pour avoir les valeurs en hexadécimal les unes derrière les autres.

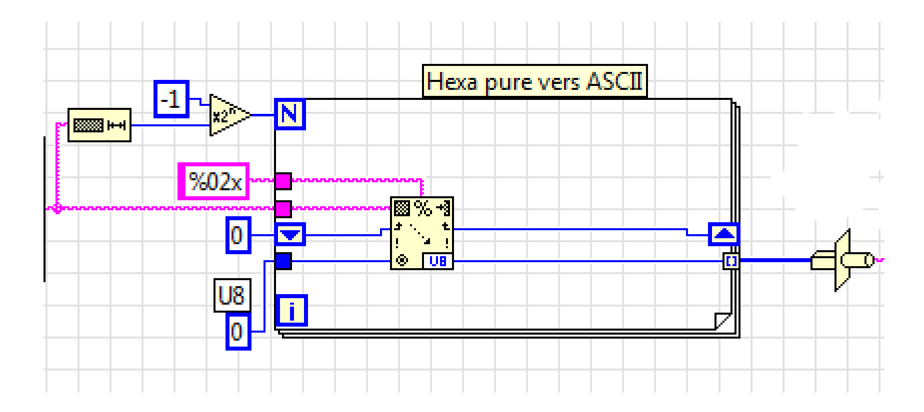



Figure 3 – conversion hexa/ASCII

Sur la figure 3 on voit comment on passe des valeurs hexadécimales en valeurs ASCII. Car le transfert par les xBee™ se fait en ASCII.

Ensuite on rajoute l'entête de notre trame soit **@G** pour les gains. On affiche les trames sur le panneau principal.

On envoie la trame quand on actionne le booléen. On envoie une trame par cinq seconde.

## Réglages des offsets des servomoteurs

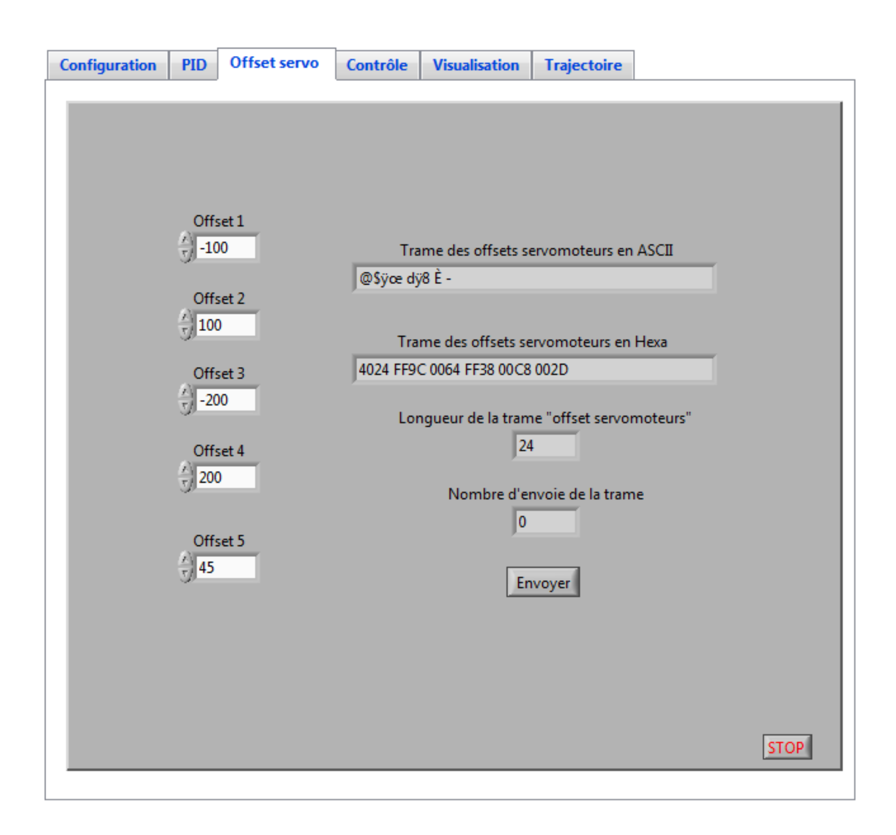



Figure 4 – écran offsets servomoteurs

Le réglage des offsets servomoteurs se fait comme pour l'envoie des gains. Sur la figure 4 on voit la page de contrôle on a les même problèmes que pour l'envoie des gains.

## Explication sommaire du code

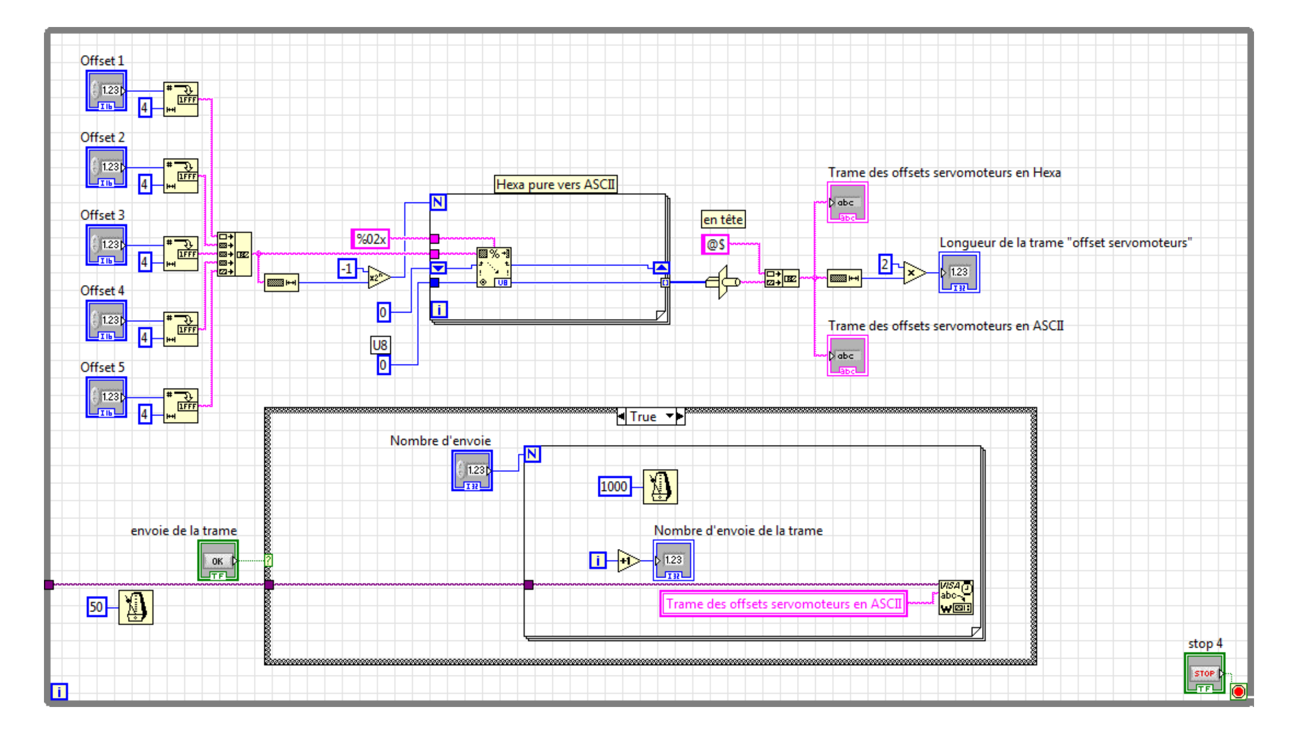



Figure 5 – Langage G offsets servomoteurs

Le code est, au sens large, le même que pour l'envoie des gains.

# Acquisition des données

L’acquisition des données se fait par le port série de l’ordinateur utilisant la norme RS-232. Nous utilisons le port série pour sa grande facilité de programmation.
Nous pouvons toujours utiliser un adaptateur Série/USB si l’ordinateur hôte n’en possède pas.
L’acquisition des données se fait dans un intervalle de temps régulier, ici on prend 500milliseconde.

On ne peut pas faire une acquisition continue car on risque un débordement de buffer. C’est pour cela qu’à la fin de l’acquisition nous vidons le buffer.

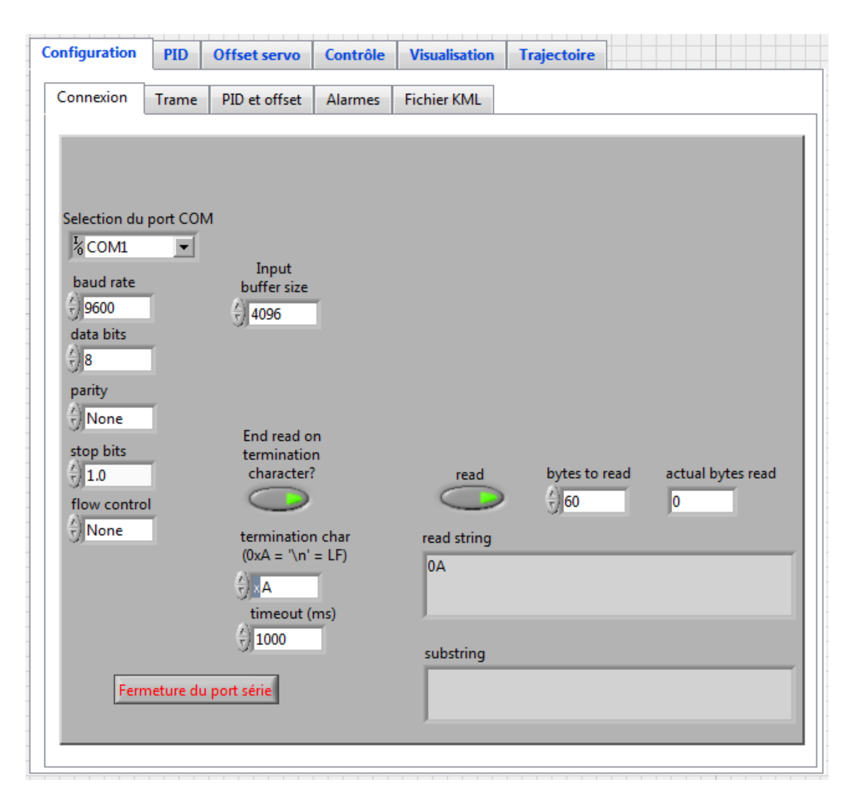



Figure 6 – configuration du port série

Sur la figure 5 on voit le panneau de configuration du port série. Pour un bon fonctionnement il faut absolument vérifier que la configuration est la bonne.

## Explication sommaire du code

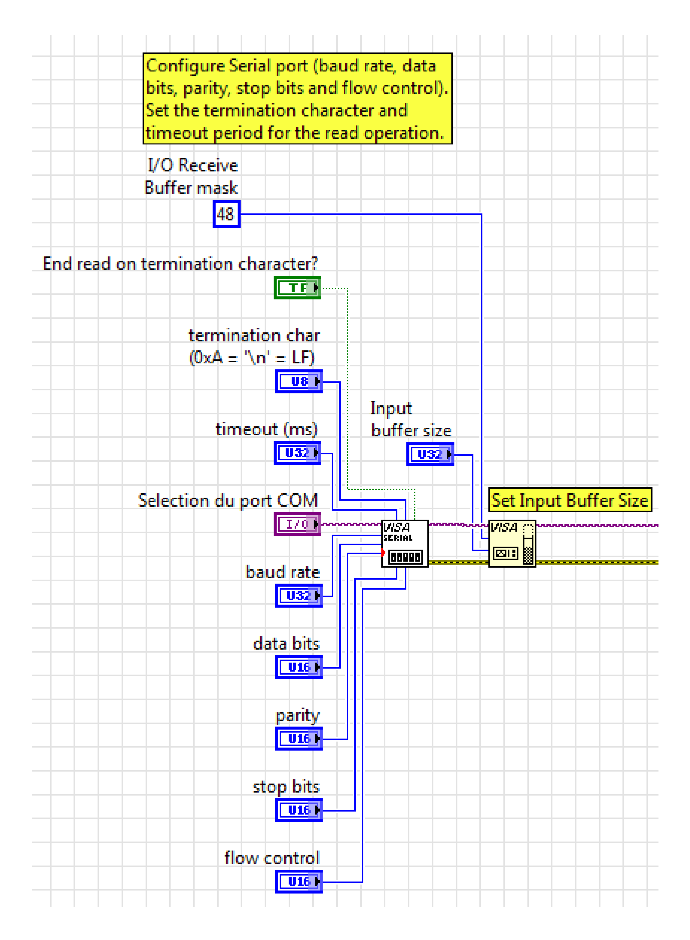



Figure 7 – code de configuration du port série

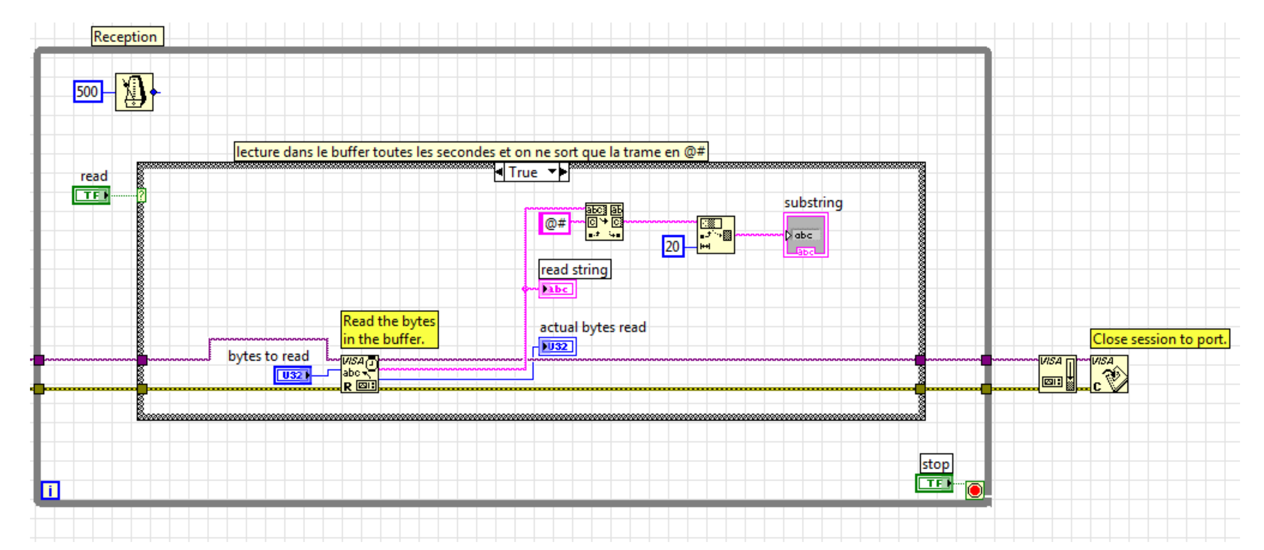



Figure 8 - acquisition de la trame

Sur la figure 7 on voit le code pour gérer la configuration du port série. On lui indique la vitesse, le numéro du port COM et toutes les autres informations obligatoires pour faire fonctionnement correctement le port COM.
Ensuite on a la partie pour le buffer. En mettant en entrée une constante qui vaut 48 cela signifie que l'on met un buffer sur en entrée et en sortie.

Sur la figure 8 on a le code d'acquisition de la trame on voit que l'on fait l'acquisition toutes les 500 msec, ensuite on lit dans le buffer sur environ 60 octets pour être sur d'avoir une trame dans ce qu'on a lu. Une trame fait 20 octets. Ce qui est lu dans le buffer est retourné sous forme de chaine de caractère.

Dans cette chaine caractères on cherche le premier entête **@#** et on extrait les 20 octets avec entête compris.

Ensuite on vide le buffer et on ferme le port série.

## Contrôle des vitesses, altitude et niveau de batterie

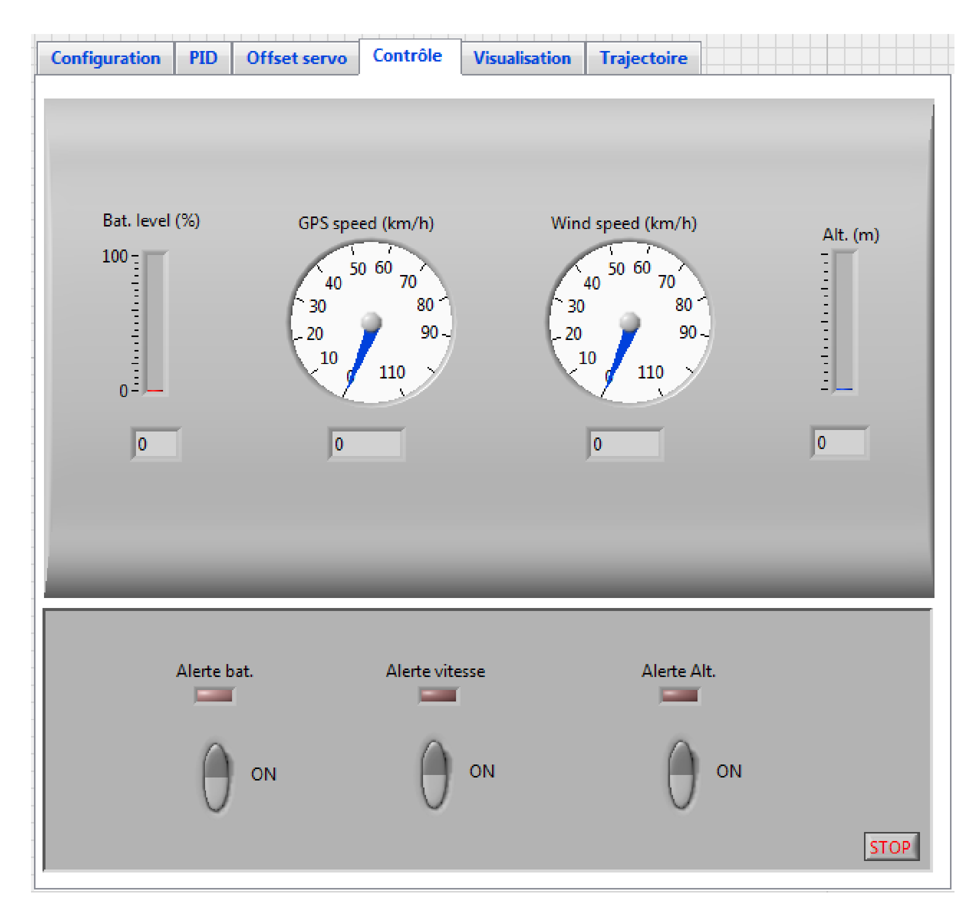



Figure 9 – écran de contrôle

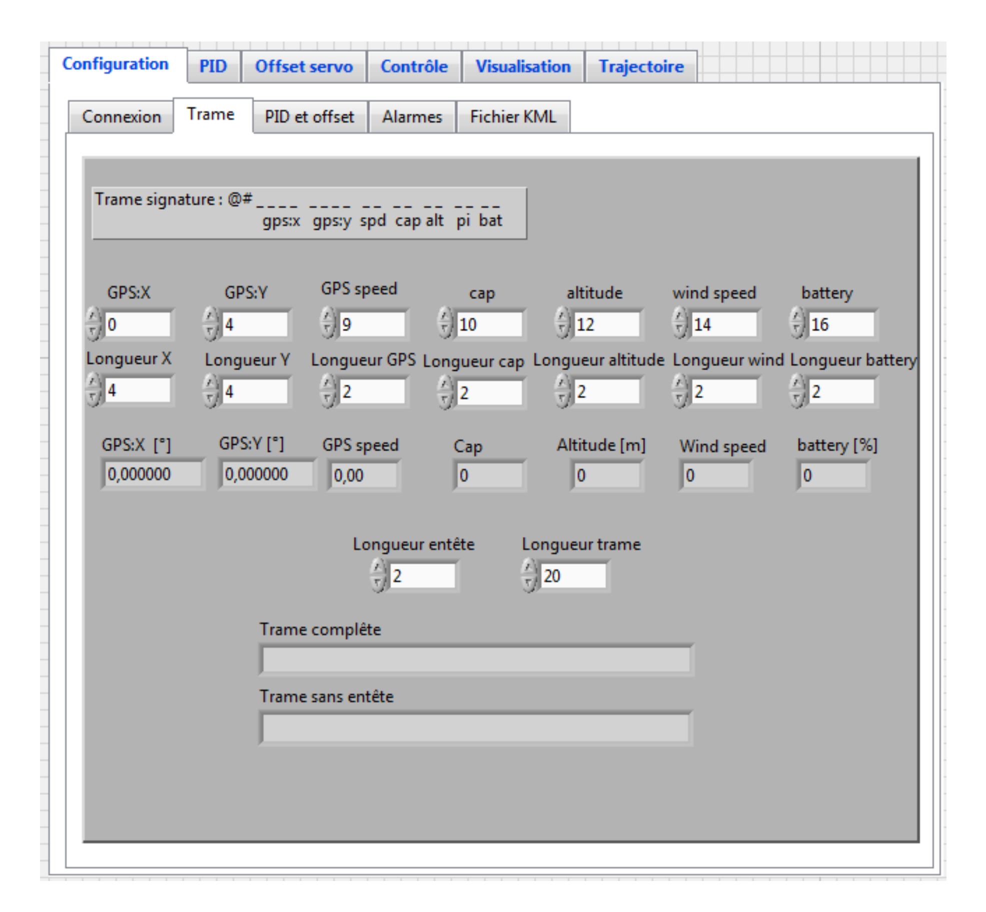



Figure 10 - configuration de la trame

Comme on peut le voir la partie contrôle gère la trame et l'affichage des différentes valeurs. On peut visualiser le niveau de la batterie en pourcentage, l'altitude, la vitesse de déplacement du GPS et la vitesse par la sonde Pitot.

Dans la partie configuration on peut configurer l'emplacement de chaque information et la longueur de celle-ci.

## Explication sommaire du code

Covnersion de la chaine ASCII en hexa pure

On multiplie par deux pour correspondre car un char = 1 octet = 1/2 chiffre hexa. Et vu qu'on a converti la trame en hexa on a besoin de deux fois plus de place....

on convertit ce que donne la trame en coordonnées gps en degrés décimaux.

Figure 11 - Ensemble du code "partie contrôle "

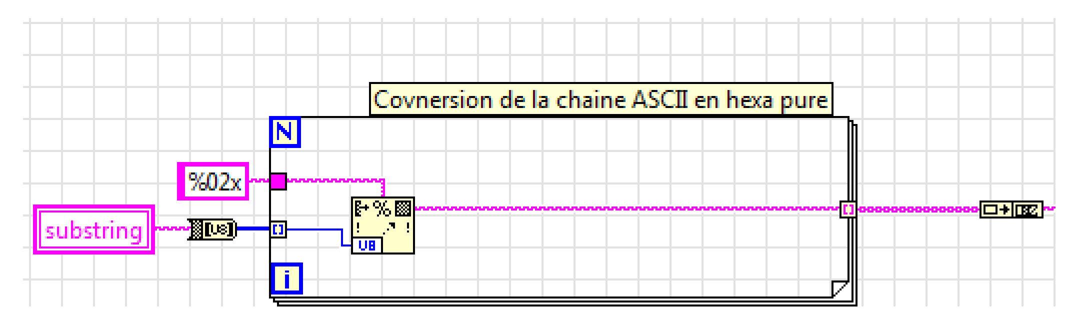



Figure 12 - conversion ASCII/Hexa

Commençons par la figure 12. On utilise la trame qu'on a extraite précédemment  sauf qu'il faut, avant de pouvoir l'utiliser, la convertir en hexadécimal car pour le moment elle est toujours en ASCII (Transfert par port série oblige).

Sur la figure 11, qui est une vue d'ensemble du code pour la partie contrôle et configuration de la trame. On peut notamment y voir les différentes conversions Hexa vers nombre décimal et la modification des valeurs GPS reçus en valeur GPS en degrés décimal.

## Visualisation de la trajectoire

La visualisation de la trajectoire, proprement dite, s'effectue sous la version libre de Google Earth©.

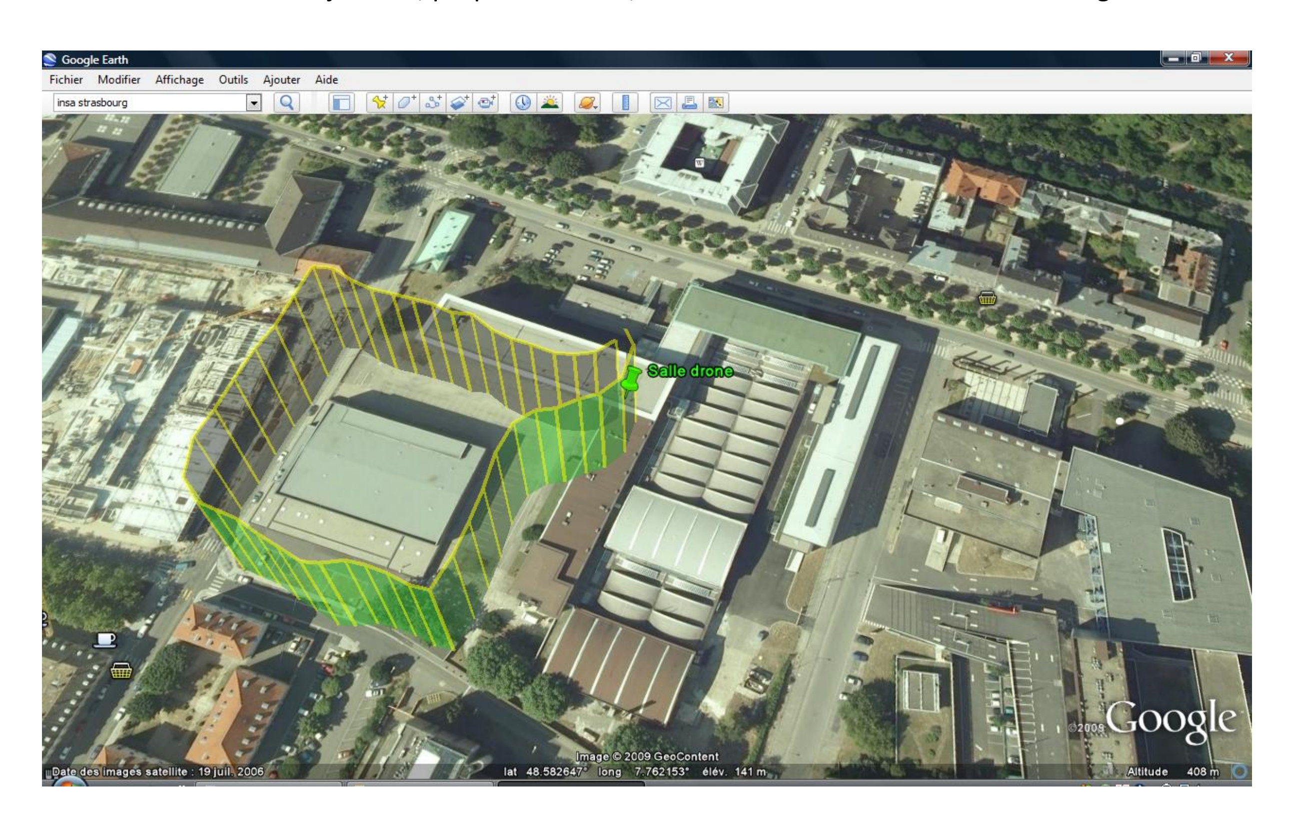



Figure 13 - Visualisation sous Google Earth©

Ci-dessus on peut voir une trajectoire près de l'INSA.
Les données sont actualisées toutes les secondes et pour ce faire on utilise la fonctionnalité réseaux du logiciel et des fichiers *.kml spécifiquement modifiée pour réaliser cette fonction.

De plus sous Labview™ on peut contrôler la position de la camera, la position actuelle, et voir différentes information sur la configuration de Google Earth©.

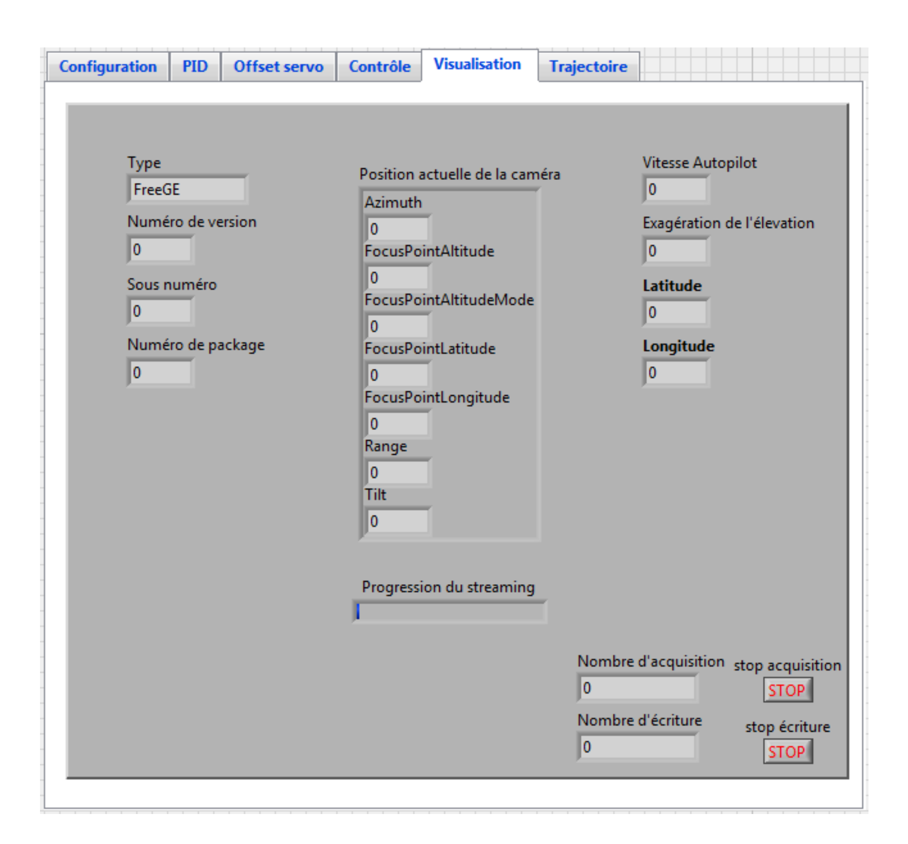



Figure 14 – Contrôle sous labview™

On a aussi un suivi du streaming des informations.

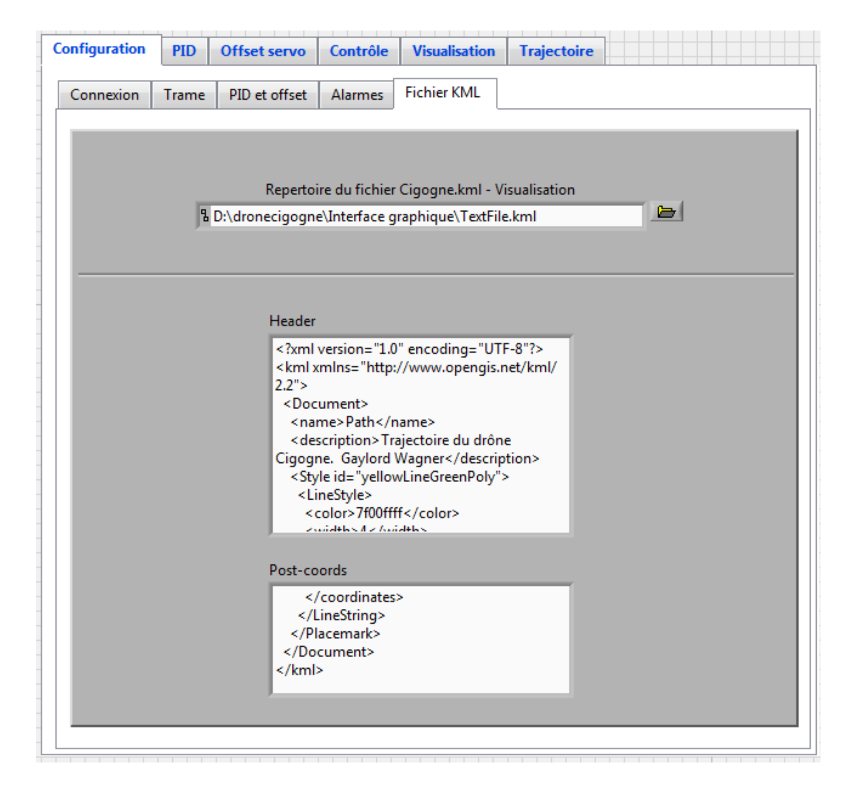



Figure 15 – Configuration

Sur la figure 15 on peut voir la configuration des entêtes des fichiers *.kml et de la sélection du fichier de trajectoire.

## Explication sommaire du code

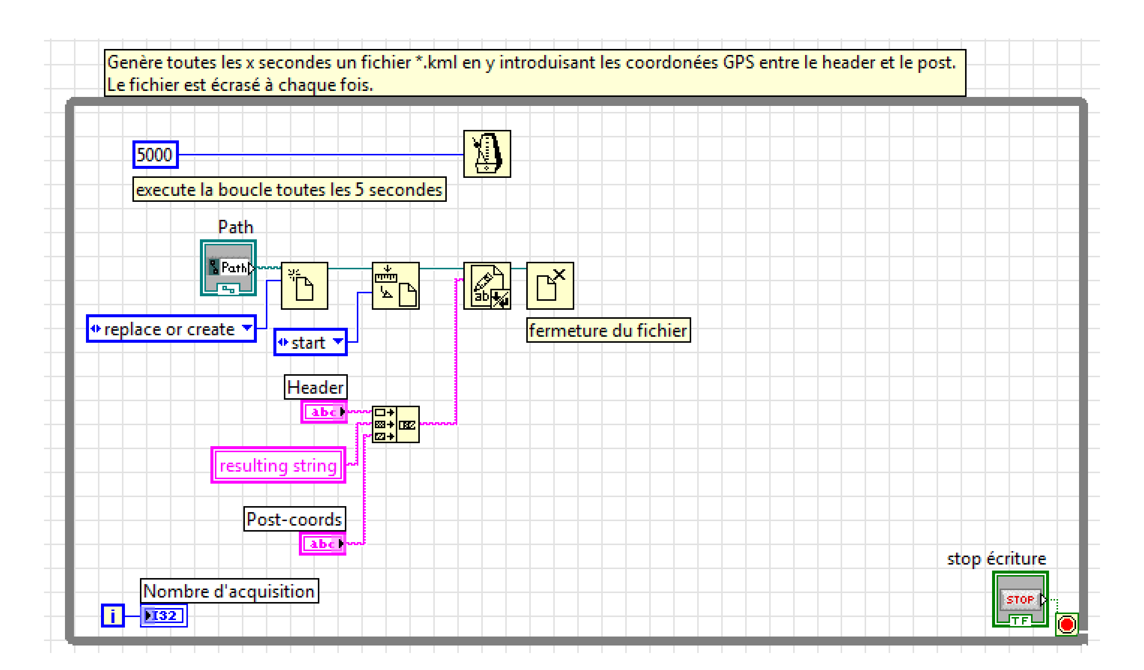



Figure 16 – création d'un fichier contenant les points de trajectoire

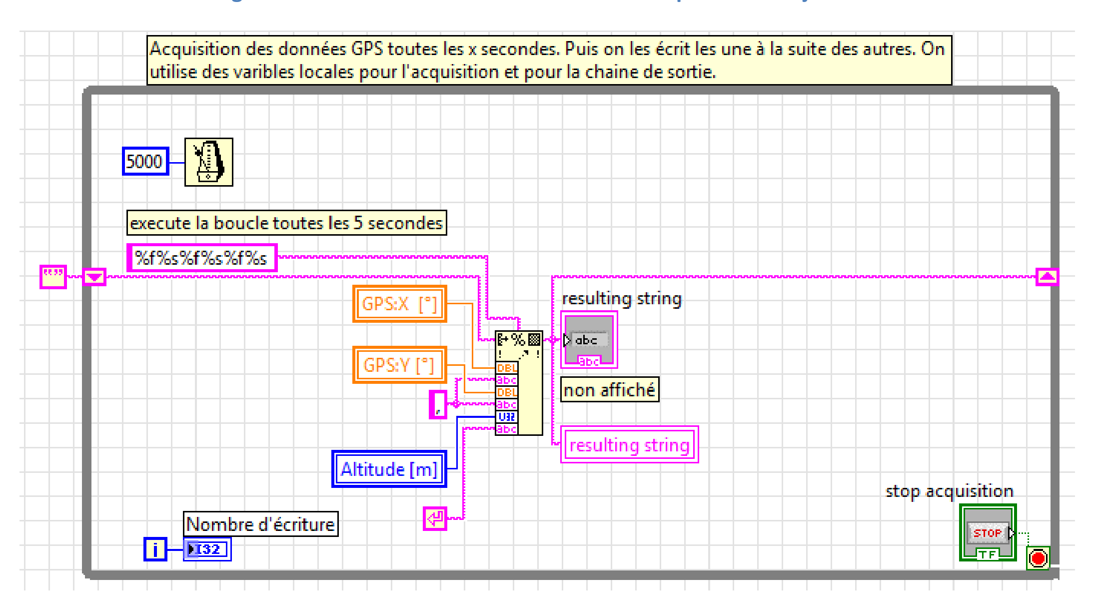



Figure 17 – Acquisition des différentes coordonnées GPS

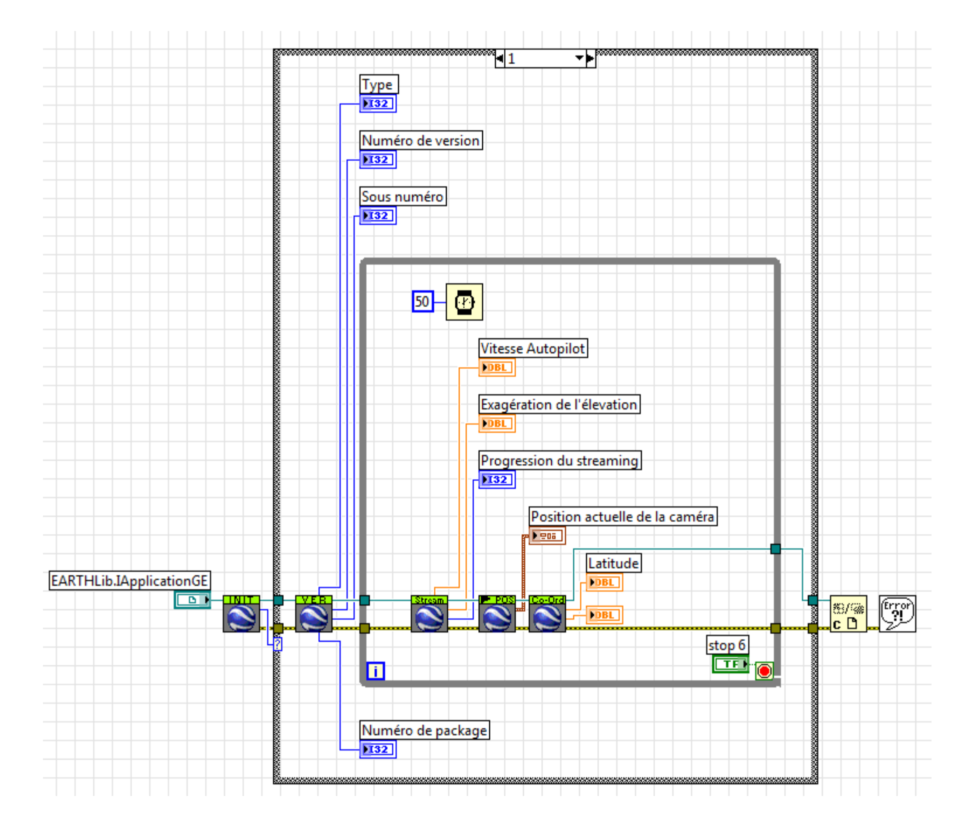



Figure 18 – contrôle et initialisation de Google Earth©

Sur la figure 16 on peut voir le code de création du fichier kml. On y voit que l'on concatène plusieurs chaines de caractères. La première, dite header, qui contient les informations nécessaires à Google Earth pour ouvrir et interpréter le fichier, la seconde qui est nos coordonnées GPS, et la dernière, dites le footer ou post-coords, qui permet de fermer les balises ouvertes.

Pour avoir les coordonnées sous la forme 7.764416965100651, 48.58186880577762, 38 j'utilise le code de la figure 17 qui me cherche les différentes valeurs et me les écrits sous cette forme pour ensuite écrire chaque ligne les unes en dessous des autres (à chaque exécution de la boucle while). Exemple :

7.764336789274173,48.58182812990097,38
7.764260192629626,48.58182094907615,38
7.764184370174272,48.5818248329985,38
7.764094384039448,48.58185261494876,38
7.764018338604918,48.58187028110072,38

Qui correspond à 5 points de positions relevées à intervalle régulier.

# *Conclusion*

Ce pré rapport a pour but de montrer une vue d'ensemble du travail déjà effectué et fonctionnel.

Une description plus précise sera faite lors du rapport final. De plus un manuel d'utilisateur et une FAQ sera faite par la suite.

L'avantage d'un travail sous Labview™ est de pouvoir avoir un code facilement éditable et compréhensible. De plus le fait de faire une visualisation sous Google Earth© nous permettra par la suite de porter les algorithmes fait sous Labview™ dans un code en C.